The Book Review Column[1]

by William Gasarch
Department of Computer Science
University of Maryland at College Park
College Park, MD, 20742
email: gasarch@cs.umd.edu

In this column we review the following books.

1. **A Computational Introduction to Number Theory and Algebra** by Victor Shoup. Reviewed by Jonathan Katz. This book focuses on the computational aspects of number theory and algebra — e.g., presenting algorithms for various tasks and analyzing their complexity — and emphasizes important applications of the mathematics developed.

2. **Primality Testing in Polynomial Time** by Martin Dietzfelbinger. Reviewed by Jonathan Katz. This book presents most of the algorithms for primality testing leading up to, and including, the result in the title.

3. **Introduction to Coding Theory** by Juerjen Bierbrauer. Reviewed by William M. Springer II. This book presents coding theory and includes the basic math, such as finite fields, that are needed.

4. **Codes: The Guide to Secrecy from Ancient to Modern Times** by Richard A. Mollin. Reviewed by Adam Bender. This is a text on cryptography that also covers some of the history and older approaches.

5. **Computational Complexity: a Quantitative Perspective** by Marius Zimand. Reviewed Carlos A.S. Oliveira. This book contains topological and measure theoretical properties of the complexity classes discussed.

6. **Secure Communicating Systems : Design, analysis, and implementation** by Michael R A Huth. Reviewed by Maulik A. Dave. The book is a textbook on security of communicating systems.

7. **Alfred Tarski: Life and Logic** by Anita Burdman Feferman and Solomon Feferman. Reviewed by Pierre Lescanne. This book covers, as the title says, both the life and logic of Alfred Tarski. Is this relevant to computer science? Read the review to find out.

Books I want Reviewed

If you want a FREE copy of one of these books in exchange for a review, then email me at gasarchcs.umd.edu

Reviews need to be in LaTeX, LaTeX2e, or Plaintext.

[1] © William Gasarch, 2006.

Algorithms and Data Structures

1. *Algorithmic Design: Foundations, Analysis, and Internet Examples* by Goodrich and Tamassia.
2. *Design and Analysis of Randomized Algorithms: Introduction to Design Paradigms* by Hromkovic.
3. *Combinatorial Optimization: Packing and Covering* by Cornuejols.
4. *An Introduction to Data Structures and Algorithms* by Storer.
5. *Handbook of Algorithms for Wireless Networking and Mobile Computing* edited by Boukerche.
6. *Solving Polynomial Equation Systems II: Macaulay's Paradigm and Grobner Technology* by Mora.

Cryptography and Security

1. *Complexity Theory and Cryptography: An Introduction to Cryptocomplexity* by Rothe.
2. *Elliptic Curves: Number Theory and Cryptography* by Larry Washington.
3. *Cryptography* by Beutelspacher.
4. *Cryptographic Applications of Analytic Number Theory: Complexity Lower Bounds and Pseudorandomness* by Shparlinski.
5. *Cryptography and Computational Number Theory* edited by Lam, et al.
6. *Coding, Cryptography, and Combinatorics* edited by Feng, Niederreiter, Xing.
7. *Foundations of Computer Security* by Salomon.
8. *Coding for Data and Computer Communication* by Salomon.
9. *Foundations of Logic and Mathematics: Applications to Computer Science and Cryptography* by Nievergelt.

Coding Theory

1. *Introduction to Coding Theory* by van Lint.
2. *Error Correction Coding: Mathematical Methods and Algorithms* by Moon.
3. *Block Error-Correcting Codes: A Computational Primer* by Xambo-Descamps.
4. *Coding Theory: A First Course* by Ling and Xing.
5. *Algebraic Coding Theory and Information Theory: DIMACS Workshop* Edited by Ashikhmin and Barg.

Game Theory

1. *The Game's afoot! Game Theory in Myth and Paradox* by Alexander Mehlman.
2. *An Introduction to Game-Theoretic Modelling* by Mestertson-Gibbons.

Combinatorics

1. *A Course in Combinatorics* by van Lint and Wilson.
2. *Weighting the Odds: A Course in Probability and Statistics* by Williams.
3. *A Beginners Guide to Discrete Mathematics* by Wallis.
4. *A Beginners Guide to Graph Theory* by Wallis.
5. *Graphs and Discovery: A DIMACS Workshop* Edited by Fajilowicz et al.
6. *Combinatorial Designs: Constructions and Analysis* by Stinson.
7. *Ordered Sets: An Introduction* by Schroder.

Logic

1. *Logicism Renewed: Logical Foundations for Math. and C.S.* by Gillmore.
2. *Essay on Constructive Mathematics* by Edwards.

Misc

1. *Domain Decomposition Methods– Algorithms and Theory* by Toseli and Widlund.
2. *The Elements of a Computing Systems: Building a Modern Computer from First Principles* by Nisan and Schocken.
3. *Handbook of Computational Methods of Integration* by Kyther and Schaferkotter.
4. *Polynomials* by Prasolov.
5. *The Political Mapping of Cyberspace* by Crampton.
6. *An Introduction to Difference Equations* by Elaydi.
7. *Difference Equations: From Rabbits to Chaos* by Cull, Flahive, Robson.
8. *Diophantine Analysis* by Steuding.

Review[2] of

A Computational Introduction to Number Theory and Algebra

by Victor Shoup

Cambridge University Press, 2005, 534 pp., $55.00

Reviewed by Jonathan Katz

Dept. of Computer Science, University of Maryland

Increasingly, number theory and algebra have become useful tools for the well-rounded computer scientist. Historically, of course, number theory and algebra have been indispensable for cryptography, and finite fields (and, to a more limited extent, other algebraic structures) have been widely used in coding theory. More recently, though, these topics have begun to permeate other areas of theoretical computer science: finite fields and error-correcting codes are now pervasive in complexity theory; quantum computing relies heavily on abstract algebra (especially linear algebra) and coding theory; and, somewhat surprisingly, there have been recent explicit constructions of combinatorial objects such as extractors which rely on deep results from number theory.

While there are numerous textbooks covering number theory, abstract algebra, and/or the basics of finite fields, there have not been many introductory-level books to approach the subject matter from a "computer science" perspective. A typical algebra textbook, for example, is more interested in proving that a certain function is *well-defined* or that a certain algebraic structure *exists* than in examining the *efficiency* of evaluating a particular function or concrete *algorithms* for generating certain objects. And certainly few books on number theory go beyond the most basic applications (if any applications are presented at all). That is not to say that there are no advanced books treating these topics; there certainly are. But the student looking for an introduction to these areas has traditionally had to learn about them from the mathematicians' perspective.

Victor Shoup, a cryptographer and computational number theorist, has changed that with his publication of *A Computational Introduction to Number Theory and Algebra*. This is an outstanding and well-written book whose aim is to introduce the reader to a broad range of material — ranging from basic to relatively advanced — without requiring any prior knowledge on the part of the reader other than calculus and mathematical maturity. That the book succeeds at this goal is quite an accomplishment! Besides focusing on the computational aspects of number theory and algebra — e.g., presenting algorithms for various tasks and analyzing their complexity — the book emphasizes important applications of the mathematics developed. Indeed, the stated purpose of the book is to cover enough mathematics to understand the applications while covering some additional material to whet the reader's appetite; as stated in the preface, the author has:

> ...tried to strike a reasonable balance between, on the one hand, presenting the absolute minimum required to understand and rigorously analyze the applications, and on the other hand, presenting a full-blown development of the relevant mathematics. In striking this balance, I wanted to be... economical and concise, while at the same time, I wanted to develop enough of the theory so as to present a fairly well-rounded account, giving the reader more of a feeling for the mathematical "big picture."

[2]©2006, Jonathan Katz

Though I might quibble here or there with a few choices, I think the author has indeed found the right balance for this book. Proofs are given for (almost) all statements, and the level of mathematical rigor and depth of material covered is high. On the other hand, applications are stressed throughout, and the presentation aims to be concrete and specific (again, with an eye toward the eventual application) rather than abstract and overly general.

The first two chapters of the book cover basic properties of the integers (introducing notions of divisibility and primality, greatest common divisors, and unique factorization) and modular arithmetic/congruences. Though the treatment here is relatively standard, it moves at a brisk pace, with ideals (in the ring of integers) already introduced in Chapter 1. Chapter 2 treats such topics as the Euler phi function, Fermat's little theorem, and Möbius inversion (which is not used in the remainder of the text). Chapters 3 and 4 illustrate the computational focus of the book, as they describe algorithms for computing efficiently on large integers (i.e., those with hundreds or thousands of digits) and for calculating the greatest common divisor, modular inverses, and more. I could continue to list the contents of the book, but (happily!) the readers of this review can obtain this information for themselves by viewing the entire book on-line at `http://shoup.net/ntb`. (Yes, that's right, the complete book is available for free from Shoup's homepage.) I will merely pique readers' interest by noting that the book also covers (among numerous other topics) basic probability theory, commutative rings, primality testing, quadratic residuosity, modules and vector spaces, matrix algebra, theory of polynomials, and sub-exponential algorithms for factoring and computing discrete logarithms.

Coming full circle to the start of my review, I would mention what some might possibly view as the only "drawback" of the book: as might already be evidenced by the list of topics above, the primary applications considered in this book are in the field of cryptography, and scant (or no) attention is given to other areas (most notably coding theory). Personally (as a fellow cryptographer), I view this as a strength of the book and, frankly, it might have been too disjointed had it tried to cover too much. But there may be some who will lament the fact that their favorite aspect of computer science is missing.

On the whole, this book is a must-read for anyone interested in computational number theory or algebra and especially applications of the latter to cryptography. I would not hesitate, though, to recommend this book even to students "only" interested in the algebra itself (and not the computational aspects thereof); especially for computer science majors, this book is one of the best available introductions to that subject.

Review[3] of
Primality Testing in Polynomial Time
by Martin Dietzfelbinger
Springer-Verlag, 2004, 147 pp., $34.95

Reviewed by Jonathan Katz
Dept. of Computer Science, University of Maryland

In August, 2002 Manindra Agrawal, Neeraj Kayal, and Nitin Saxena posted an astonishing paper on the Web with the simple title: "Primes is in P." The AKS paper was

[3]©2006, Jonathan Katz

astonishing not only because it resolved a widely-studied question that had been open for roughly 30 years (or 200 years, or 2000 years, depending on how one chooses to count such things), but also because the techniques used in the paper were almost completely elementary, relying almost entirely on standard undergraduate algebra. The paper caused quite a stir within the computer science community, and within days (even hours!) of its posting numerous researchers had already verified the proof, given seminars on the result, and suggested improvements and simplifications (to the point where the proof now relies entirely on undergraduate algebra alone). Due to its fundamental nature, its accessibility, and its relevance to a wide range of disciplines, it is fair to say that the paper is probably the most widely read paper in theoretical computer science of the past 5 years.

The meaning of the result stated in the title is that it is possible to decide in deterministic polynomial time whether a given number n is prime or composite (here, the running time is measured as a function of the length of the binary representation of n). It is worth remarking that the result has little practical significance: randomized polynomial-time algorithms for the same problem (with exponentially-small probability of error) have been known and widely used since the late 1970s, and the deterministic algorithm given in the AKS paper is much slower than these; furthermore, the techniques do not currently seem to shed any light on the problem of factoring n efficiently, should it be determined that n is not prime. From a theoretical point of view, however, the paper resolved a fundamental open question and moreover did so using beautifully simple techniques.

The aim of *Primality Testing in Polynomial Time*, by Martin Dietzfelbinger, is to lead the reader through a full proof of the AKS result without assuming much more than mathematical maturity on the part of the reader. (Indeed, though the reader who has never seen group theory before might find the book tough going, all the algebra necessary for the proof is developed from scratch in the book.) It is a testament to the simplicity of the AKS result that such an accomplishment is feasible.

The book can logically be separated into two parts: the first covering introductory material and the second covering the AKS result itself. Chapters 1–7 (as well as an Appendix) contain the introductory material. Chapters 1 and 2 motivate the problem and cover the basics of algorithmic notation and complexity theory. These chapters are intended for mathematicians, as an undergraduate computer science major should be comfortable with such material. Chapter 3 describes results and terminology from basic number theory, including the Euclidean algorithm for finding the greatest common divisor of two numbers, modular arithmetic, the Chinese remainder theorem, and the density of prime numbers. This chapter is intended for computer scientists, as an undergraduate mathematics major should be comfortable with this material. Chapter 4 moves on to modern algebra, introducing groups and subgroups, the notion of cyclic groups and generators, as well as rings and fields. Chapter 7 covers polynomials over rings, roots of polynomials, and other related topics. As an interlude between Chapters 4 and 7, Dietzfelbinger describes the randomized primality testing algorithms that were developed in the late 1970s, and have become essential for modern applications of cryptography: Chapter 5 describes and analyzes the Miller-Rabin test, while Chapter 6 covers the Solovay-Strassen algorithm. Along the way, these chapters introduce additional necessary algebraic concepts, including quadratic residuosity and the Jacobi/Legendre symbols.

Chapters 1–7 are a joy to read, and I found the proofs and explanations clear and concise. Amazingly, the material is presented in full, with complete proofs given for all

results necessary for proving the main results of the book. As an unavoidable consequence, this means that the book is not a substitute for a course in modern algebra — anything not needed for a subsequent result is simply omitted — but also means that an undergraduate (or possibly even a motivated high-school student) could sit down and verify the complete proof, from scratch, that primality testing can be done in polynomial time.

The reader of this review may note that Chapter 8, which contains the actual proof of the AKS result, was not included in the glowing review above. I simply found this last chapter to be too terse, especially given that it is the *raison d'être* of the book. The chapter is only 15 pages long, and though the proof was clear and understandable — and I was able to verify everything, step-by-step — when I finished the book I was left with no intuitive feel as to *why* the proof "worked." I also feel that our diligent undergraduate/high-school student, who has been breezily following along until this point (if the act of reading mathematical proofs can be said to be "breezy"), is likely to become frustrated upon reaching the last chapter and, though she will get through it, will certainly have to struggle to do so. I would have preferred that the author had doubled the length of this chapter by including more motivation, additional explanations, and (especially) some toy examples along the way to make the concepts more clear.

I did not find any serious errors in the book (though there were a few typos), but I did find some problems with the bibliography: a paper by Agrawal and Biswas from 1999 was an important precursor to the AKS result, but is not cited here; also, some URLs listed in the bibliography are no longer valid.

Who would benefit from this book? Note that the AKS paper itself, as well as some later expository papers written by Dan Bernstein, are freely available on the Internet and are *almost* as easy to read as Chapter 8 of this book (though, ultimately, the proofs in this book are more accessible). Thus, for someone already comfortable with all the algebra covered, it would be hard to justify buying this book; I should note, though, that even such a reader would benefit also from coverage of the randomized primality testing algorithms in Chapters 5 and 6. On the other hand, I would enthusiastically and wholeheartedly recommend this book for the student who may have forgotten some algebra (or who may never have learned it in the first place), but who is interested in learning about the proof of the AKS result; I would, however, forewarn them that they will have to persevere in order to get through the final chapter.

Review of[4]
Introduction to Coding Theory
Author of Book: Juergen Bierbrauer
Publisher: Chapman and Hall/CRC, 2005, ISBN 1584884215
Review by William M. Springer II (`wmspringer@gmail.com`)

1 Overview

What is coding theory? In its essence, coding theory can be broken down into three main components: secrecy (the encrypted message should give an enemy as little information as possible), efficiency (coding/decoding should be fast and the encrypted message should take

[4]©2006, William M. Springer

up as little space as possible) and redundancy (it should be possible to detect and correct errors in the transmission). In many applications, interception is not a concern, so the focus is on transferring information quickly with little possibility for error.

In a perfect world, codes could be used to transmit information with zero redundancy, allowing the transfer of information with maximum efficiency. In practice, noisy channels introduce the need to be able to detect and correct transmission errors; the challenge is to provide maximum redundancy with minimal waste. In its simplest form, redundancy can be provided by simply transmitting every message several times; fortunately, far more efficient mechanisms are available.

This book is part of the Discrete Mathematics and its Applications series, and as such is heavily mathematical; however, most of the required math is presented in the early chapters; a basic grounding in linear algebra should be enough to follow the presentation. More complex structures such as finite fields and linear codes are defined as they are encountered.

2 Contents

Part I, An Elementary Introduction to Coding, introduces the reader to basic concepts such as linear codes, hashing, entropy, and projective planes. This section, around 130 pages, gives the reader a firm grounding in the terminology and ideas needed for part II. Other concepts in this section include hamming distance, block coding, and 3-dimensional codes.

Chapter one starts out by defining such concepts as bits, binary addition (or XORing), the finite field F_2, and error correction. Hamming distance is introduced and used to measure the distance between binary strings; a code with eight codewords having minimum distance three is presented, and it is shown that any one error in transmission can be detected and fixed. Probability is used to determine the number of expected errors in a given transmission.

Chapter two begins the introduction of concepts from linear algebra, such as dimension, generator matrices, linear independence, and determinants. A linear binary code is defined to be a binary code which is closed under addition; this allows a code C to be specified by a generator matrix which generates all codewords of C.

Chapter three expands into general linear codes, rather than just the binary linear codes considered before. Prime fields and congruences are introduced, followed by the process of constructing a finite field. The game SET is introduced, where completing a SET is identical to finding a valid codeword, and previously developed techniques are applied to the analysis of the game. Finally, decryption speed is increased through the use of cosets and syndrome decoding.

Chapter four is short, focusing on Reed-Solomon codes, which are maximum-distance separable codes, meaning that the codewords are at the maximum possible distance from each other.

While the first four chapters focused on constructing codes from scratch, chapter five looks at extending and shortening existing codes. In particular, the Reed-Solomon codes discussed in chapter four are extended by concatenation.

Chapter six and seven are both quite short; chapter six introduces hash functions and their application to error detection, while chapter seven describes cyclic codes and the binary Golay code.

In chapter eight, we learn about Shannon entropy, one of the key components of information theory. Entropy is a measure of uncertainty; it describes the randomness of a system. Maximum entropy is obtained when all outcomes are equally likely, while entropy is zero if the outcome is predetermined (and thus grants no new information) A fair coin, for example, would have entropy of $\frac{1}{2}$, while a double-headed coin would have entropy of zero (as the outcome will always be heads).

Chapters nine and ten round out the first section of the book with some more advanced concepts including asymptotic bounds, three dimensional codes, and projective planes, while chapter eleven introduces the next section of the book.

Part II, Theory and Applications of Codes, starts by introducing more properties of finite fields, then moves on to cyclic codes, geometric codes, quantum codes, and various applications.

Chapter twelve starts off part two, which is quite a bit more complex than the first half of the book. This chapter introduces the Galois group, trace and subfield codes, and the duel code.

In chapter thirteen we return to cyclic codes, constructing them over the field F_{16}. There is a discussion of the application of binary linear codes in fingerprinting, which allows a pirated document to be traced back to its source. Generator and check matrices of cyclic codes are discussed, and the Euclidean algorithm is brought into play.

Chapter fourteen continues the discussion of recursive construction of codes started in chapter five. Covering codes are defined and related to steganography, the science of hiding both the message and the existence of the message.

Chapter fifteen is on the use of orthogonal arrays in statistics and computer science. This chapter also introduces cryptography, the art of sending secret messages. The DES cryptosystem is briefly explained, and references are given to more modern cryptosystems such as AES/Rijndael. Resilient functions are introduced, with an application in foiling wiretapping. Also mentioned are stream ciphers, one time pads, and Monte Carlo algorithms. Several pages are devoted to authentication, efficiency, and privacy amplification.

Chapter sixteen is devoted to the geometric description of codes. Projective geometry is explained, and linear codes are described as multisets of points on the projective plane. Much of the chapter is devoted to quadratic forms and caps (points on a projective plane such that no three points are collinear).

Chapter seventeen covers additive codes. Applications in computer memory systems and deep-space communication are briefly discussed, and a section is devoted to quantum codes.

The final chapter focuses on boundaries, including linear programming bounds, sphere packing, and the kissing number (a variant of sphere packing). The last few sections are devoted to nonlinear codes.

3 Opinion

I enjoyed reading this book; the author is good about defining terms as he uses them so that the logic can be followed by someone without a background in the area. The sections are short, with relevant problems at the end of each section. Answers to the problems,

however, are not provided. While the book gets very technical, particularly in the later chapters, it should be accessible to the typical computer science graduate student; I would recommend it for a graduate class on coding theory. Students without as much experience following dense reasoning, however, may prefer an easier-to-read text such as Trappe and Washington's Introduction to Cryptography and Coding Theory.

Review of[5]:
Codes: The Guide to Secrecy from Ancient to Modern Times
by Richard A. Mollin
Chapman & Hall/CRC, 2005, 679 pp., $79.95, Hardcover

Reviewed by Adam Bender
Dept. of Computer Science, University of Maryland

1 Overview

Richard Mollin has written many books ([2, 3]) on cryptography for readers at many levels. I found this to be his most approachable work, and one of his most informative. At first I expected this book to be similar, in that it would focus mostly on crypto. I was surprised to see that book covers a wide range of topics, some of which are not directly related to cryptography. It can more properly be described as the history and detail of many security primitives and protocols. It would be a good book for a beginner to read straight through (perhaps flipping back and forth to the appendices) in order to get a solid foundation in the fields of cryptography and security. As a general reference however, it falls somewhat short.

2 Summary of Contents

This book can be cleanly divided into three sections, which focus on history, cryptography, and computer security.

Chapters 1 and 2 present a history of cryptography from its inception to the beginning of the "modern era," when public-key crypto was developed. Just about every development in the history of crypto is at least mentioned. The author provides lengthy historical details in order to set up the background for each development in the history of cryptography. Often the history outweighs the crypto, for instance, on the two pages about Mary, Queen of Scots, crypto is mentioned for just half of a paragraph. This is not to say that the material is uninteresting, just that someone interested solely in crypto might want to skim this section. The only thing lacking in these chapters was a thorough treatment of the creation and cryptanalysis of the German WWII-era Enigma cipher, which is one of the greatest cryptanalytic feats in history and lead to the development of the modern computer.

I will discuss the appendices here, because they should be read before reading the next sections, especially chapter 4. Appendix A, "Mathematical Facts", covers basic number theory, abstract algebra, matrices and vectors, with a touch of elliptic curves and complexity theory. Hardly any proofs or motivation are given, and the material is barely comprehensive

[5] ©2005 Adam Bender

enough to support some of the topics presented in the book. Yet there is probably enough to allow this book to be considered as self-contained. Further appendices deal with elementary probability theory, as well as algorithms for factoring, primality testing and discrete log. Descriptions of AES and PRNGs are also included.

Chapters 3 to 5 deal with actual cryptography. Chapter 3 on symmetric ciphers is rather well done, with some more history (bios of Schneier and Rivest) thrown in. In particular there is a good explanation of S-DES, a miniature DES algorithm, that helps explain Feistel networks and block ciphers in general. Stream ciphers are also addressed.

Chapter 4, Public-Key Cryptography, presents RSA, digital signatures, and ElGamal, as well as their underlying problems: discrete log and integer factorization. It presents many of the concerns that arise when dealing with PKC, especially security issues and specific attacks. This book does a better job than most at introducing the basic attacks and explaining why "vanilla" protocols are not secure. This is also the first chapter in which a familiarity with mathematical notation and number theory is assumed. A reader unfamiliar with number theory should go over the appendices until they have a good understanding of at least Appendix A, as there are few references to the appendices to guide a struggling reader.

It is here that some of the claims break down as well. Up until now the only thing I felt was inaccurate in the book, which goes out of its way to dispel commonly held inaccuracies, was that its definition of cryptography was overly complex, to the point where it might not be technically correct. But on page 168, the author claims that given a one-way function f, and nonce n, the least significant bit of the sequence $f(n+i)$ for i increasing from 0 is a PRNG. This is false, as the least significant bit of f could be constant without affecting the one-way-ness of f. This error also appears in other works as well, such as [7], thus highlighting the need for rigorous proofs of all non-obvious material. As far as I can tell, this is the only significant error in the book.

Common cryptographic protocols are presented in chapter 5. These give further motivation for and applications of cryptography. Specific topics presented include oblivious transfer, identification schemes, bit commitment, secret sharing, electronic voting, SSL, and digital cash. Again, the treatment is very thorough and includes many details and background information. This is the final chapter that I feel deals mainly with cryptography, although it could be argued that the next two chapters do as well.

More full-fledged protocols are presented in chapter 6, Key Management. Key authentication, exchange and distribution, as well as PKI and Secure Electronic Transmission (SET), are presented here.

Heavy math notation returns in chapter 7 on Message Authentication. Here many types of authentication methods are discussed, including hash functions (along with the birthday attack), message authentication codes, HMAC, combining authentication with encryption and applications of authentication.

Here, the book diverges almost completely from the topic of cryptography, and begins to cover general security. Chapter 8, Electronic Mail and Internet Security, talks a great deal about PGP and S/MIME, and also about firewalls and browser cookies. These topics are not typically found in a book about crypto, and instead are in the realm of computer security.

Chapter 9 proceeds with a presentations of passwords, biometrics and nuclear test ban treaty compliance. Chapter 10 is entitled Noncryptographic Security Issues, and includes

sections on viruses and copyright issues. In these sections I encountered many things that I felt were extraneous, including, for example, a note on K&R style indenting of C code, the meaning of the cracker term "leet" (short for elite), the popularity of Nutella vs. peanut butter, and how Time Warner came into existence. While all of these facts might be enjoyable to some, for me they just seemed to get in the way. This is part of why I feel this book is better read straight through, as a long story about the history and development of security issues.

The final chapter, on information theory, returns to a mathematical theme, but there isn't much of an explanation for why a reader interested in cryptography should also be concerned about this as well. Given the subject of many of the latter chapters, I think a more appropriate subtitle would be **The Guide to Communication Security from Ancient to Modern Times**.

3 Conclusion

The intended audience of this book is different for each section. The first two chapters could be read by, and would probably be of interest to, just about anyone. The remainder of the book is written for someone at the undergraduate level. The more technical chapters would be useful to a math student, while the implementation details seem geared toward computer science and information technology students.

The appendix of exercises makes this book good for a self-guided introduction to cryptography and protocols, similar to [4] but more in depth and up-to-date. As far as its use as a textbook, I feel that this book would better serve as a primer or supplementary text for a computer security course than a stand-alone crypto text. It is math- oriented, but not very many types of math are used (mostly number theory), and hardly any proofs are presented. For a rigorous treatment of cryptography, one should refer to [1] or [6]. This would be a good book for the crypto portion of a network security course, akin to [5].

Important notes are spread throughout the book, so it is probably best to read it once from cover to cover to get a good introduction to cryptography itself and many of the related topics. The lack of proofs and concise definitions hamper its use as a reference, although its thorough survey of security-related subjects makes it a good introduction to the field.

References

[1] Oded Goldreich. *Foundations of Crypography: Volume I, Basic Tools*. Cambridge University Press, 2001.

[2] Richard A. Mollin. *An Introduction to Cryptography*. Chapman & Hall/CRC, 2000.

[3] Richard A. Mollin. *RSA and Public-Key Cryptography*. Chapman & Hall/CRC, 2002.

[4] Simon Singh. *The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography*. Anchor, 2000.

[5] William Stallings. *Cryptography and Network Security: Principles and Practice*. Prentice Hall, 2002.

[6] Douglas Stinson. *Cryptography: Theory and Practice.* Chapman & Hall/CRC, 2nd edition, 2002.

[7] Lawrence C. Washington and Wade Trappe. *Introduction to Cryptography with Coding Theory.* Prentice Hall, 2002.

1

Review of[6]
Computational Complexity: a Quantitative Perspective
Author of Book: Marius Zimand
Publisher: Elsevier, ISBN 0444828419, 352 pages, Hardcover
Author of Review: Carlos A.S. Oliveira[7]

1 Introduction

The area of computational complexity has nowadays a good number of introductory books at the graduate level. On the other hand, it is not so easy to find comprehensive and well written books on more advanced topics of complexity. It is often the case that, after the first graduate course in computational complexity, the only avenue for new explorations of the theory is to go directly to papers in the area. This is in some respects unfortunate, because there is no easy transition from introductory books, which have to operate with several simplifications (required for the understanding of new readers), to the real-world practices of the research community. The book "computational complexity: a quantitative perspective" offers a different learning path, by combining several important results, among others that are not so well known, and presenting a selection of topics that are more challenging than that provided in the average textbook on complexity theory.

The basic premise of the work is to present a quantitative view of computational complexity results. By this, the author means the study of topological and measure theoretical properties of the complexity classes discussed. However, the end result is a book that not only introduces quantitative results, but provides also a review of several important topics in complexity theory that are often relegated to the optional chapters in traditional textbooks.

2 Book Contents

Preliminaries. The introductory chapter provides the basic notation and definitions used in the book. Turing machines (deterministic and nondeterministic) are introduced, the concept of computability is discussed, along with the notion of computational complexity classes. Another important part of the chapter is the presentation of the measure theoretical concepts used in quantitative results.

Measure theory is concerned with assigning a measure to sets in such a way that they can be meaningfully compared to each other. Starting from notions of topology such as open sets and neighborhoods, the book progresses to the application of these concepts to

[6] ©2006 Carlos A.S. Oliverira
[7] School of Industrial Engineering and Management, Oklahoma State University. 322 Engineering North, Stillwater, OK 74078. Email: coliv@okstate.edu

complexity classes. A topological space is a set X together with a collection $\mathcal{O}$ of subsets of X, which are called *open sets*. A neighborhood of $x \in X$ is any open set containing x. A topological space can be classified as *nowhere dense*, *first Baire category*, *second Baire category*, *co-meager*, and *co-nowhere dense*. A nowhere dense set A is one in which for each open set U_1 there is an open set $U_2 \subseteq U_1$ such that $A \cap U_2 = \emptyset$. Intuitively, a nowhere dense set is one in which there are many holes; in other words, we can always find open sets which are not included in A in any given region of X. Other classifications simply improve this notion: a first Baire category set is one that can be represented as a countable union of nowhere dense sets. Sets that are not of the first Baire category are of the second Baire category. Co-meager and co-nowhere sets are just the complements (in X) of meager and nowhere dense sets.

Using this terminology, one can define a topological space based on strings of symbols. Such spaces can be used to study topological properties of classes of computational complexity languages. Given the binary alphabet $\Sigma = \{0, 1\}$, and a lexicographic ordering of the countably infinite set of strings in Σ^*, any language Λ can be represented by an infinite string in Σ^∞ where the i-th character is 1 if and only if the i-th string in the aforementioned enumeration is in Λ. Therefore, the set of all languages in Σ^* can be seen as the set of infinite strings in Σ^∞, and we set $X = \Sigma^\infty$ in the definition of the topological space.

The open sets of X can be defined in two ways. In the *Cantor topology*, the collection $\mathcal{O}$ of open sets is countable, with one open set for each string $s \in \Sigma^*$. Each open set U_s^C is defined as the set of all infinite strings in Σ^∞ where the sequence of first $|s|$ characters is identical to s. The *superset topology* is defined similarly, but here the open sets U_s^S are collections of infinite strings in Σ^∞ where, for the first $|s|$ characters, the i-th position is one whenever the corresponding position of s is also one. Intuitively, an open set in the superset topology corresponds to all possible extensions of a given language.

Abstract Complexity. The second chapter delves into the notion of abstract complexity and associated topological properties. By *abstract complexity theory* it is meant the analysis of resources in a generic way, through the use of the *Blum axioms*. Given an enumeration $\varphi_1\varphi_2\ldots$ of the set of partial computable functions (formally known as a gödelization of the set of partial computable functions), the abstract complexity measure of a function φ_i is a partial function Φ_i (the resource function) such that (1) the function Φ_i is defined whenever φ_i is defined, and (2) for all values $x, y \in N$ it can be computed if the resource function $\Phi_i(x) \leq y$. The requirements (1) and (2) are known as Blum axioms. A sequence of pairs of partial functions (φ_i, Φ_i) where $\varphi_1\varphi_2\ldots$ is a gödelization and each pair satisfies axioms (1) and (2) is known as a *Blum space*. In this setting, a computational complexity class can be defined relative to a resource (given by the partial functions Φ_i) and a resource bound g (which must be a computable function). The computational class is denoted as C_g^Φ and represents all partial functions using at most $g(x)$ units of the resource defined by Φ, for any given input x.

One of the main results of the chapter is the demonstration that any complexity class, when considered as a topological structure, is small compared to the set of computable functions. For this to make sense when talking about computable functions, the definition of nowhere dense sets is slightly modified to make sure we are considering only open sets corresponding to computable functions. A second interesting result related to abstract complexity theory says that, for any resource bounding function g, the size of the set of

computable functions requiring more than $g(x)$ units of resource on input x is not small. The proof of this result relies on the speedup theorem, which says that there are computable functions (say f) with the following property: given any algorithm for f, it is always possible to find another algorithm to compute f using less resources.

A second interesting result in abstract complexity theory is the gap theorem. In this theorem, we are given a function F such that $F(f, x)$ is much bigger than $f(x)$, for any partial computable function f and input x. The theorem then says that it is always possible to find a function f such that $C_f^\Phi = C_{F(f)}^\Phi$, for some resource function Φ. In other words, given any increase in resources (determined by F) it is still possible to find problems for which this increase of resource does not translate into an increase of computational power. From the quantitative (topological) point of view, the important observation is that the set of such problems is not small.

The classes P, NP, and EXP. The third chapter explores the connections between the classes P, NP and EXP. Initially, the idea of finding probabilistic algorithms with complexity $p(n)(3/4)^n$, which is correct with probability at least $1 - 1/e^n$ for some polynomial function p. The second issue addressed in the chapter is the topology of the set NP $\setminus$ P, assuming P $\neq$ NP. The topological space considered in these results is derived from the superset topology on the set of computable functions, as defined in the first chapter. It is shown that in this case NP $\setminus$ P as defined in the first chapter. can be classified as a second Baire category set (which, from the definition, means that it is not a small set). On the other hand, it can be proved that assuming NP $\neq$ P the class of NP-complete problems is only a first Baire category set, and therefore it can be expected that many problems are neither in P nor NP-complete. Similar results are shown using measure-theoretic concepts. Under this point of view, it can be shown that the set P has measure zero, while EXP has measure greater than zero. Thus, there is a clear separation between the two sets in terms of measure theory, which suggests the question: does the set NP have measure zero or greater than zero? – given the results above, proving that the measure of NP is non-zero will essentially prove that NP $\neq$ P. The chapter concludes with a discussion of average-case complexity issues.

Quantum Computation. The study of quantum algorithms has attracted the interest of many researchers, due to the potential of this area for providing a new paradigm of computation. The discovery of quantum algorithms for difficult problems such as integer factoring shows that quantum computers could be more powerful than their classical counterparts. A good introduction to quantum algorithms is provided in the fourth chapter. A quantum machine is one where quantum bits (also known as qbits) are manipulated. Contrary to classical machines, whose bits can have only zero or one values, qbits can have any value that is a linear combination of the "standard" qbit positions $|0\rangle$ and $|1\rangle$ (which perform the same role as zero and one in a classical computer). The main difference between a quantum and a classical system is that while a qbit can be in any state that is a combination of $|0\rangle$ and $|1\rangle$ (in fact combinations must also form a unitary vector), only one of these two vectors are observed by any measurement, and the probability of each occurring is equal to the square of its amplitude in the current state.

Finite automata can be constructed on quantum computers in a way similar to their definition for classical systems. For this, a sequence of qbits must be manipulated (this

is usually done using tensors of qbits and unitary matrix operators that implement the transition function of the automata). An interesting result about quantum automata is that they are actually less powerful than classical automata: they cannot recognize the class of regular languages. On the other hand, quantum Turing machines look much more promising. For example, it is known that they can be used to find the prime factors of any integer number. The book presents a proof of this superiority for a different problem that has the following property: it can be solved in polynomial time by a quantum computer with black-box access to an oracle, but it requires exponential time on a classical computer with black-box access with black-box access to the same oracle, for almost all inputs. This is proved, as was to be expected, using measure-theoretical techniques.

One-way Functions and Pseudo-random Generators. Chapter five is dedicated to computational complexity concepts that are most useful in cryptography applications. In particular, one way functions and pseudo-random generators are studied. Several interesting results are presented but, differently from the previous chapters, these results do not rely heavily on topological and measure-theoretic techniques. This fact makes the chapter a little less appealing for the audience interested mainly on quantitative aspects. On the other hand, readers with an interest in complexity results in general will find a nice introduction to the main ideas of one-way functions, pseudo-random generators, and extractors.

Optimization Problems. Another important application of computational complexity concepts occurs in combinatorial optimization. In these problems, finding the optimum value of a function is essential, and therefore approximation becomes one of the main techniques. Chapter 6 presents ideas used to solve these problems, including approximation algorithms. In order to understand the inherent limitations of such approximation procedures, a logical characterization of NP is presented. This leads to a logical description of optimization problems, in terms of formulas of first order logic with a specific ordering of quantifiers – different orderings generate optimization problems that correspond to different complexity classes, as proved in the chapter. In fact, it is shown that there is a relationship between the form of the logical formulas representing optimization problems and classes of the polynomial hierarchy.

Chapter 6 also discusses the classification of optimization problems into classes, according to their approximation properties. Examples of such classes are given for minimization and maximization problems. Finally, the chapter discusses the notion of probabilistic checkable proofs, which lead to a new characterization of NP. The PCP theorem is discussed, and a weaker version of the theorem is proved. This important result is then used to provide lower bounds on approximation for problems such as Clique and Set Cover.

3 Conclusion

One feature of this book makes it very easy to spot what is going on in each chapter and section, and should be adopted by other authors: most sections are introduced with a one paragraph, informal summary, starting with the words "IN BRIEF". This is an excellent help to anyone that is trying to quickly find some specific information, without the required time to read a whole section. It is also instructive for readers that are seeing the material

for the first time and want to know what a section is all about, before spending a good deal of time reading it. Finally, it sometimes works simply as a motivation to continue reading. In a similar vein, many theorems are started with an "informal statement", followed by a "formal statement" with all technicalities added.

The book provides not only a guided tour through quantitative issues in computational complexity, but it is also a good introduction to some advanced topics. This is notably true for quantum computation and cryptography, pseudo-random generators and expanders. The author exercised a good balance between intuitive arguments, provided as informal descriptions, and exact formalism. I think this is very satisfying for the reader, since it gives proper context for the discussion. Moreover, the author does not make the mistake of being sloppy with notation, as occurs on some textbooks. All proofs in this book are complete, and are given with good contextual explanations.

In summary, this is a really good book in advanced topics of computational complexity. It is easy to read despite the heavy notation, which is well explained. It can be very useful for people interested in the quantitative aspects of computational complexity. But even if this is not the case, the book may serve as a well thought introduction to some modern topics in complexity theory.

Review of[8]

Secure Communicating Systems : Design, analysis, and implementation
Author of book: Michael R A Huth
Published in 2001 by Cambridge University Press, 283 pages

Reviewer: Maulik A. Dave

1 Overview

The book is a textbook on security of communicating systems. Security is one of the major concerns while transferring data on communicating systems like Internet. This textbook presents topics like public-key cryptosystems, stream and block ciphers, some secure communication protocols, and related topics. It contains a detailed description of an advanced encryption standard (AES) of NIST, the cipher Rijndael, announced as winner of the AES design competition on October 2000. Various aspects of communication include communication between two individuals, communication involving public, and communication in programs. The book describes both theories, and practices. The theories include definitions, lemma, proof of theorems, correctness of algorithms, examples, and the necessary mathematical background. The practices include various protocols, various related algorithms, methods used in real life securities, and corresponding programs in pseudo code.

2 Content Summary

The book is divided into 6 chapters. The first chapter is on general introduction. The next two chapters are on cryptography. The last three chapters are on security of information flow.

Chapter 1 talks about socio-economics aspects of the security, and can be skipped during first reading. Chapter 2 is on public-key cryptography. Taking RSA as an example, it covers specification, realization, correctness, security, and the combinatorial aspects of public-key cryptography. The Miller-Robin algorithm is described.

In the last subsection, Diffie-Hellman Key-Exchange System, Station to station protocol, Massey-Omura Cryptosystem, and ElGamal cryptosystem are briefly introduced. Chapter 3 describes symmetric-key cryptography. It covers stream ciphers, and block ciphers. Apart from Rijndael, it introduces various other ciphers. Chapter 4 is on security protocol design and analysis. It covers digital signatures, secure log-ins, secret sharing, and model checking protocols. Chapter 5 revisits the public-key encryption with RSA, discusses efficiency of implementation of RSA. Chapter 6 is on analysis of secure information flow. The analysis is being explained with the type theory, and the theory of semantics. A small language is presented for explaining the relational semantics. The chapter ends with a small introduction of program certification in the context, and a small introduction to covert channels. The book also has an appendix to present some necessary fundamentals of Combinatorics.

[8]©2006 Maulik Dave

3 Conclusion, and Comments

The book can be a good text book at undergraduate level, and graduate level. It has a large number of exercises at the end of sections, and many subsections. This makes the textbook useful for the students. Each chapter has a section in the end, containing bibliographic notes. Students wanting to read more on the topics can use the 3 pages of bibliography supplied in the book. The author M. Huth is Senior lecturer at Imperial college London. His experience is reflected in the writing style.

To understand the book, preliminary knowledge of computer internals, algorithms, and number theory is required. The number theoretic results, required to understand the theories, are presented in the book. The book is well written to be suitable for self study. The researchers, and professionals in the area of security, may find the book useful as a reference.

Review of[9]
Alfred Tarski: Life and Logic
Author of Book: Anita Burdman Feferman and Solomon Feferman
Cambridge University Press (2004) (v+425pp).
Reviewed by Pierre Lescanne

Ones first contact with this volume, even before opening it, is the beautiful cover portrait of the young Tarski due to the half mad painter Stanisław Witkacy. It shows all the fever of the character. As soon as one gets the book, one wishes to know more.

One may wonder why the life of a logician is relevant for a computer scientist. The question is even more appropriate when one discovers that this scientist, who felt himself along his life close to mathematicians and philosophers, had no specific attraction toward the emerging computer science, despite what he was told by people like George Collins and Michael Rabin of the potential interest of these applications. The reason why the biography is of interest for computer scientists is that, despite Tarski's wish or anticipation, his influence on computer science is definitive. Well known are the results on decidability, in general, and on decidability of geometry and real closed fields that led to Collins' algorithm and its application to computer algebra and automated deduction. But there is a more surprising result that I learned reading this book: his works on cylindric algebras anticipated those of Edgar F. Codd on relational calculus for data bases. It is worth recalling that as a pioneer in formal semantics, he was the forerunner of semantics of programming languages. Is not Dana Scott one of his former students? If one likes general history and science history, Tarski's biography is fascinating and Anita Burdman Feferman and Solomon Feferman paint it with warmth and life. The account is well built and holds the reader spellbound. For instance, at some point one may be surprised by a digression or an occurrence of characters and one does see at first why they are relevant; only later one understands their role. At some points, the book moves the reader, as Tarski's life was full of ordeals which he had to endure.

In addition to having an exceptional scientific stature, that made him with Kurt Godel to dominate mid twentieth century logic, Alfred Tarski was a personage, a true character, a hard worker with an iron constitution that allowed him to resist to the conjunction of

[9] ©2006 Pierre Lescanne

tobacco, alcohol, pills, sleepless nights. In a memorial speech his long friend and colleague Leon Henkin described him with an alliterative series of adjectives: "proud, penetrating, persistent, powerful, passionate". Reading this book one understands that the character was contrasted: articulate, authoritarian, charming, clear, demanding, hard worker, imperial, macho, selfish, sociable, good speaker all together. He made his contemporaries, his students and his co-workers feel an amazing mixture of repulsion-adoration. As an illustration, I have heard myself, even in France and even nowadays, activist anti-tarskists despite the fact that three universities (one in France) gave him a honorary degree of Doctor Honoris Causa: Calgary, Santiago de Chile and Aix-Marseille, while his homeland Poland and his second homeland the United States did not. Professor Tarski or "Papa Tarski" as his students used to call him was also a womanizer with much attraction for female persons working in his surroundings.

Beyond Tarski's life, the book leads us into History with a description of the prewar Polish academic and cultural world which I hardly knew. It gives also unknown facets of logicians and philosophers of which I was happy to know more. To cite only a few (this is a subjective list): Carnap, Ehrenfeucht, Ershov, Kripke, Montaigue, Presburger, Quine, Scott. But this books introduces us into Logic. Indeed, six crystal-clear interludes present large parts of logic.

This biography allowed me to understand how much Tarski influenced contemporary science, from mathematics to human sciences through biology and computer science of course. This is explicit in his involvement in the *Unity of Science* movement, but not only. I discovered the influence he had on our own research. Clearly, I recommend reading *Alfred Tarski: Life and Logic* to all computer scientists, theoreticians or not, passionate about history and history of science or not, as we all need to better understand our field and its emergence.